Utilización de ADA y VHDL para el Codiseño Hardware Software de Sistemas de Tiempo Real, modelado mediante Redes de Petri

H. H. Mazzeo
FI- UNLP
CeTAD
La Plata, Arg.
hhmgvm@yahoo.com

J. A. Rapallini
FI- UNLP
CeTAD
La Plata, Arg
josrap@gmail.com

W. Aroztegui
FI- UNLP
CeTAD
La Plata, Arg
walter.aroztegui@gmail.com

A. A Quijano
FI- UNLP
CeTAD
La Plata, Arg
adrian.quijano@gmail.com

J. M. F. Ocampo
FI - UNLP
CeTAD
La Plata, Arg
jmfocampo@ciudad.com.ar

***Resumen* -- Se presenta un método didáctico para el diseño e implementación de sistemas de tiempo real (STR). Dada la especificación del STR, los alumnos trabajan en su modelado con Redes de Petri, con el lenguaje ADA para generar la simulación funcional y con VHDL para llegar a la implementación final realizada con herramientas de diseño comerciales. Para ilustrar el método se muestran dos aplicaciones con sistemas empotrados y según la partición realizada a partir de los conceptos de Codiseño Hardware-Software (CoHS), se utiliza lógica programable para su resolución**

Palabras claves: ***VHDL, ADA, Sistemas de Tiempo Real Codiseño Hardware/Software***

I. INTRODUCCIÓN

La existencia de una amplia variedad de ambientes de software permite una rápida especificación, simulación, verificación, realización de modificaciones, correcciones, optimizaciones e implementaciones de un sistema.

Para el caso de desrrollo de un STR, la utilidad de las Redes de Petri corresponde a la modelación de sistemas de tiempo real [1]. El hecho de ser una herramienta gráfica simple le otorga practicidad para la representación de sistemas complejos con diferentes grados o niveles de abstracción. Desde el punto de vista didáctico, presenta amplias ventajas para la enseñanza ya que no se requiere prácticamente ningún conocimiento específico de hardware o software para el análisis de estos sistemas [2].

La utilización de ADA [4] [5] y VHDL [6] [7] para la especificación e implementación de un sistema son utilizados en esta propuesta metodologica. Ambos son capaces de cubrir la mayoría de las tareas de codiseño software-hardware.

Las posibilidades que ofrece el lenguaje ADA para la confección de programas multitarea, manejo de tiempos y concurrencia, hacen que sea uno de los más adecuados para la representación de sistemas de tiempo real. A partir de este punto, la etapa siguiente es la traslación del código obtenido a lenguaje VHDL para la implementación del sistema de control en un dispositivo lógico programable adecuado.

II. METODOLOGIA

Una vez obtenida la especificación del modelo del sistema que cumple con las condiciones impuestas e incluso considera las restricciones de implementación, es preciso generar una implementación software de la red de Petri, consistente en un programa que simula el disparo de las transiciones (es decir ejecuta las acciones asociadas a éstas) observando las reglas de evolución del marcado. Se ha elegido Ada 95 como lenguaje de programación, debido a sus especiales características de concurrencia y de tiempo real.

La obtención del código VHDL deberá seguir un proceso de traslación de ADA a VHDL. En algunos casos será directa por la semejanza en la sintaxis de ambos lenguajes. En otros, será necesario un proceso más complejo para la generación del código VHDL y en algunos casos la traslación a VHDL no será posible. Su similitud sintáctica con el lenguaje ADA, facilita la traslación del "código software" al "código hardware". A pesar de ello habrá porciones de código ADA que no serán directamente trasladables a código VHDL, en cuyo caso habrá que realizar las modificaciones o agregados de código adecuado de modo que el sistema, manteniendo la misma funcionalidad, cumpla sus objetivos.

Luego, a traves de las herramientas de software [8] [9], se verifican e implementan los diseños realizando los correspondientes prototipos funcionales.

III. CASOS DE ESTUDIO

Para ilustrar el procedimiento metodologico, se presentan dos casos de estudio.

El primero, un sistema destinado al control de la circulación de trenes sobre un circuito ferrovial. La función principal del sistema consiste en controlar la circulación de cada tren actuando sobre cambios de agujas con accionamientos eléctricos, con el fin de evitar colisiones. El monitoreo de la ubicación de los trenes se realiza mediante sensores de posición ubicados estratégicamente sobre el tendido ferrovial.

Se parte del modelado en Redes de Petri (Fig.1), pasando por el desarrollo del programa en ADA (Fig.2 y Fig.3), llegando a traves del ambiente de desarrollo Quartus II (Fig.4).

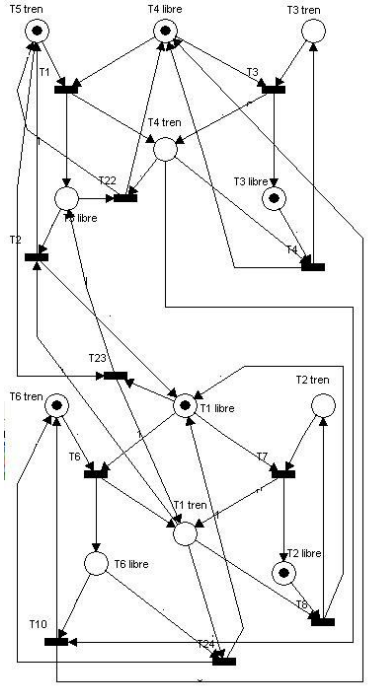

Figura 1: Modelización por medio de Redes de Petri

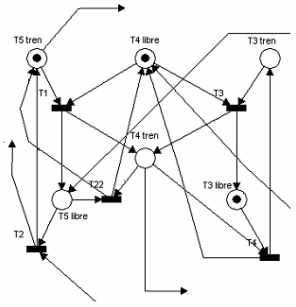

Figura 2. Estructura básica usada para la programación en ADA

```
type Sitios (T4Libre);
type Tentrada (T4, T10, T22);  -- transiciones de entrada
                               -- al semáforo T4libre
type Tsalida (T1, T3);         -- transiciones de salida
                               -- del semáforo T4libre

protected Semaforo is
    entry Marcar (Tentrada);
    entry Desmarcar (Tsalida);
private
Marca: Integer:=1;     -- marca de T4Libre
-- estructura de sitios y transiciones
-- dependiente del resto del sistema
end Semaforo;

protected body Semaforo is
    function Habilitacion (T:Tsalida) return Boolean is
        begin
            if Marca := 1 then T := TRUE;
            return T;
            end if;
        end;
            -- devuelve TRUE si T4Libre está marcada

    entry Desmarcar (for T in Tsalida) when Habilitación(T) is
        begin
            Marca := Marca - 1;
            --decrementa marcación de T4Libre
        end;

    entry Marcar (for T in Tentrada) when TRUE is
        begin
            Marca := Marca + 1;
            --incrementa marcación de T4Libre
        end;
end Semaforo;
```

Figura 3. Código en ADA de parte de la estructura de figura 2

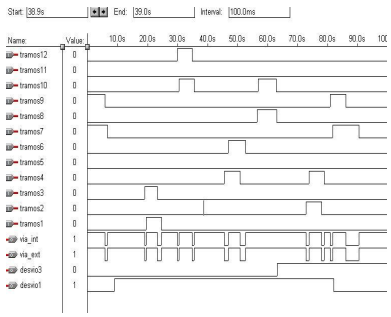

Figura 4: Verificación - Diagrama de tiempos del caso de estudio 1

Se desarrolló un modelo simple en escala tipo maqueta por el que pueden circular las formaciones ferroviarias por diferentes tramos y en distintas direcciones.(Fig.5)

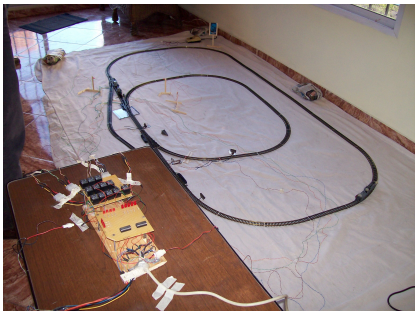

Figura 5: Prototipo funcional del caso de estudio 1

El segundo caso corresponde a la implementación de un equipo didáctico para la enseñanza de microprogramación, como el caso anterior se realizaron los pasos ya explicados llegando finalmente a al implementacion con lógica programable. La CPU realiza operaciones aritméticas y lógicas básicas con operandos de 3 bits de longitud sobre una EPLD7128 de Altera y displays de 7 segmentos para la visualización de resultados. (Fig.6)

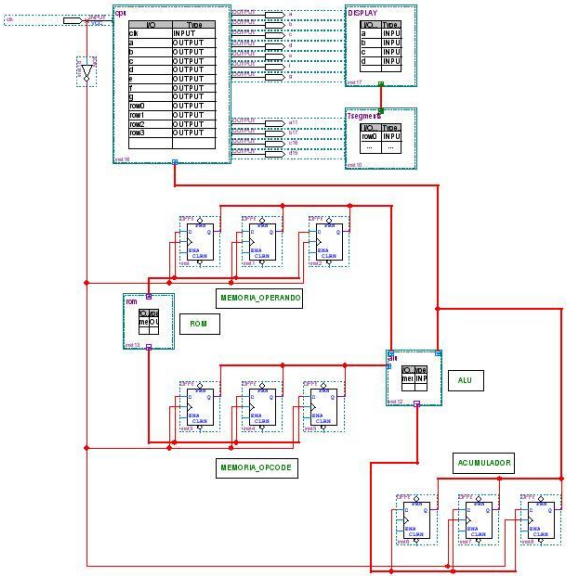

Fig. 6 Diagrama esquemático en Quartus II del caso de estudio 2

IV. CONCLUSIONES

El hecho de obtener la implementación completa del modelo propuesto, estimula el aprendizaje de todos estos temas de una forma más relacional y coordinada. La idea de partir de un modelo en papel hasta obtener un prototipo funcionando, crea un mayor grado de entusiasmo que el hecho de estudiar los temas en forma aislada y sin un objetivo claro, estimulando el desarrollo de proyectos sobre temas prácticos que de otro modo nunca llegan a implementarse.

REFERENCIAS

[1] *Redes de Petri: Modelado de Sistemas Concurrentes*. Departamento de Electrónica y Computadores. Universidad de Cantabria. Curso 2007-2008.

[2] Kubátová Hana, "*Teaching Principles of Petri Nets in Hardware Courses and Student's Projects*", Department of Computer Science and Engineering, FEE, CTU in Prague

[3] E. Villar y A. López, *"Especificación de sistemas embebidos en Sistemas digitales: elementos para un diseño a alto nivel"*, A.García, Comp. Bogotá: Programa de Ciencia y Tecnología para el Desarrollo – Ediciones Uniandes, 1999, pp. 25 – 96.

[4] Michael A. Smith, *"Object-oriented Software in ADA 95"* Second Edition. School of Computing University of Brighton.

[5] Manuel Carro, *"Concurrencia en ADA: objetos protegidos"*, Universidad Politécnica de Madrid, 20 de noviembre de 2005

[6] 1076TM IEEE Standard VHDL Language Reference Manual. 2002.

[7] Yalamanchili. "Introductory VHDL: From Simulation To Syntesis". Prentice Hall. 2001

[8] Bergeron, Janick. "*Writing Testbenchs - Functional Verification of HDL Models*". Kluwer Academis Publishers. 2000

[9] Quartus II - http://www.altera.com/products/software/quartus-ii/web-edition/qts-we-index.html